

# Navigator QOS Firmware Specification

Version A  
Mar 17 2015

## **Document License**

This work is licensed under the Creative Commons Attribution-NoDerivs 3.0 Unported License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nd/3.0/> or send a letter to Creative Commons, 171 Second Street, Suite 300, San Francisco, California, 94105, USA.

## **Contributors to this document**

Copyright (C) 2015 Texas Instruments Incorporated - <http://www.ti.com/>

| Revision Record                                      |   |
|--|---|
| Document Title: <b>Software Design Specification</b> |   |
| Revision   | Description of Change                   |
| A  | i. Initial Release (Firmware v 1.0.0.8) |
|  |   |



## **Contents**

|   |                              |
|---|------------------------------|
| <b>1. OVERVIEW</b> .....                          | <b>5</b>                     |
| <b>1.1 RELATED DOCUMENTS</b> .....                | ERROR! BOOKMARK NOT DEFINED. |
| <b>1.2 OVERVIEW</b> .....                         | <b>5</b>                     |
| 1.2.1 What this Specification Provides.....       | 5                            |
| <b>2. FUNCTIONAL DESCRIPTION</b> .....            | <b>6</b>                     |
| <b>2.1 BASIC OPERATION</b> .....                  | <b>6</b>                     |
| 2.1.1 QOS Queues.....                             | 7                            |
| 2.1.2 Egress Queues.....                          | 7                            |
| <b>3. QOS ALGORITHM DESCRIPTION</b> .....         | <b>8</b>                     |
| <b>3.1 SOFTWARE OVERVIEW</b> .....                | <b>8</b>                     |
| 3.1.1 Modified Token Bucket Algorithm.....        | 8                            |
| <b>3.2 QOS COMPONENT SPECIFICATION</b> .....      | <b>11</b>                    |
| 3.2.1 QOS Queue.....                              | 11                           |
| 3.2.2 QOS Cluster.....                            | 12                           |
| 3.2.3 RR-MODE QOS Cluster.....                    | 13                           |
| <b>3.3 SRIO QUEUE MONITORING</b> .....            | <b>15</b>                    |
| 3.3.1 SRIO Queue Configuration .....              | 15                           |
| <b>4. FIRMWARE COMMAND INTERFACE</b> .....        | <b>17</b>                    |
| <b>4.1 FIRMWARE COMMAND HANDSHAKE</b> .....       | <b>17</b>                    |
| 4.1.1 Command Handshake.....                      | 17                           |
| 4.1.2 Command Buffer .....                        | 17                           |
| 4.1.3 QOS Queue Region Base .....                 | 18                           |
| 4.1.4 Timer Configuration.....                    | 19                           |
| 4.1.5 Enable / Disable QOS Cluster.....           | 19                           |
| 4.1.6 Enable / Disable SRIO Queue Monitoring..... | 20                           |
| <b>4.2 INTERNAL MEMORY ALLOCATION</b> .....       | <b>21</b>                    |
| 4.2.1 PDSP / QMSS Scratch RAM Allocation .....    | 21                           |

# 1. Overview

---

---

## 1.1 Overview

This document specifies the PDSP firmware operation and command interface of a “Quality of Service” (QoS) functionality designed to run on the Queue Manager subsystem of Nysh.

### 1.1.1 What this Specification Provides

- Functional Description
  - Basic Operation
  - Algorithm Details
- Host processor interface
  - Firmware command interface
  - Scratchpad memory usage

## 2. Functional Description

### 2.1 Basic Operation

The quality of service (QoS) PDSP is charged with the job of policing all packet flows in the system, and verifying that neither the peripherals nor the host CPU are overwhelmed with packets.

The key to the functionality of the QoS system is the arrangement of packet queues. There are two sets of packet queues, the QoS ingress queues, and the final destination queues. The final destination queues are further divided into host queues and peripheral egress queues. Host queues are those that terminate on the host device and are actually received by the host. Egress queues are those that terminate at a physical egress peripheral device.

When shaping traffic, only the quality of service (QoS) PDSP writes to either the host queues or the egress queues. Unshaped traffic is only written to QoS ingress queues. It is the job of the QoS PDSP to move packets from the QoS ingress queues to their final destination queues while performing the proper traffic shaping in the process.

Figure 1 below is an abstract illustration of the basic concept. Note that traffic going to both the Host CPU and the Hardware Device(s) is shaped by controlling how and when the QoS PDSP moves packet from the QoS Flow Queues to their final destination.

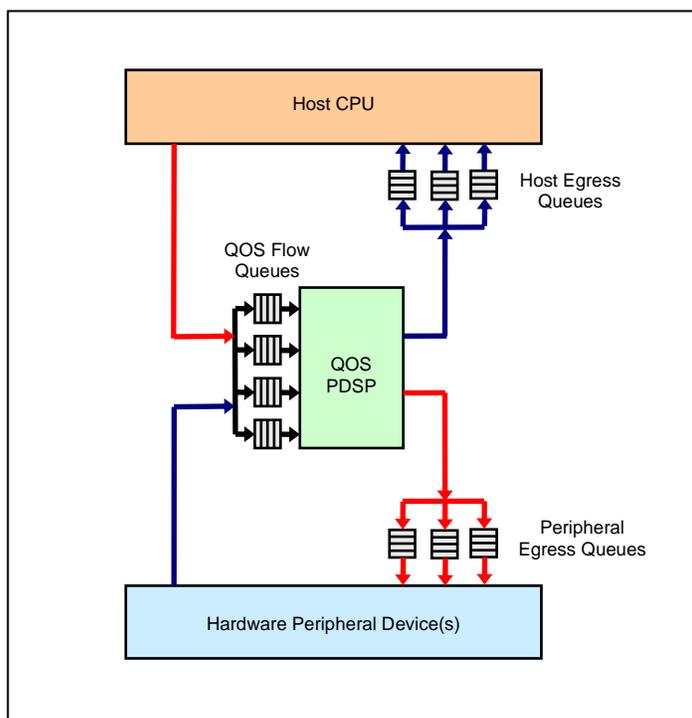


Figure 1 – Interface to QoS PDSP Engine

### **2.1.1 QOS Queues**

There is a designated set of queues in the system that feed into the QOS PDSP. These are called QOS queues. The QOS queues are simply queues that are controlled by the firmware running on the PDSP. There are no inherent properties of the queues that fix them to a specific purpose.

### **2.1.2 Egress Queues**

Egress queues are those that feed a physical egress peripheral or feed the host CPU. From the QOS PDSP's perspective, the final destination of any packet on a QOS queue is always an egress queue. Egress queues are referred to as peripheral egress queues when they feed a peripheral device, and host egress queues when they feed the host. The term "peripheral egress queue" is used more often here as this document typically assumes the QOS shaping is for transmit unless otherwise noted.

The egress queues keep track of the number of packets queued so that the QOS PDSP can get an idea of the egress devices' congestion level. Although there is no limit to the number of packets that can be placed on these queues, it is intended that they be kept shallow such that high priority packets can maintain a small latency period to their destination.

## 3. QOS Algorithm Description

---

---

### 3.1 Software Overview

The firmware assumes 64 QOS queues are allocated to the QOS PDSP. They are physically located at a fixed base (most likely not zero), but are referred to as QOS queues 0 through 63 in configuration. The fixed base has a default value that is set via an equate in the firmware, but can be overridden by a configuration command.

A QOS queue is a rate and congestion controlled channel that feeds into a single egress queue. Multiple QOS queues can merge onto a single egress queue, but each individual QOS queue may have only one destination.

The QOS firmware is designed around the idea that multiple QOS queues are group together to provide multiple flows and priorities to a single egress device. A group of QOS queues with a common egress device queue is called a QOS queue cluster (or QOS cluster).

A QOS cluster is created through the host software by first initializing the individual QOS queues used in the cluster, and then creating a QOS cluster that encompasses the queues in question. Different QOS algorithms can be executed on individual QOS clusters.

#### 3.1.1 Modified Token Bucket Algorithm

##### 3.1.1.1 Basic Operation

The modified token bucket algorithm allows each queue in a cluster to be assigned a fixed rate in bytes per time iteration (typically 25us, but is configurable). This is called iteration credit. In addition, there is a maximum number of bytes that can be retained as credit against a future traffic burst. This retained credit is called total credit. The maximum limit is called maximum credit.

Iteration credit is added to a queue's total credit at the start of each sampling period. While total credit remains above zero, the packet waiting at the head of the QOS queue is examined for size. If the byte size of the packet is less than or equal to the queue's total credit, the packet is forwarded and the packet byte size is deducted from the credit bytes. The queue's unused credit is carried over to the next iteration (held in its total credit), up to the maximum amount allocated to the queue.

For example, if a flow is rated for 40Mb/s, but can burst up to 20,000 bytes at a time, the queue would be configured as follows on a system with a 25us iteration:

Iteration Credit = 125 bytes                      *(40Mb/s is 125 bytes every 25us)*  
Maximum Credit = 20,000 bytes

The sum total of iteration credit for all queues in the cluster should add up to the total expected data rate of the egress device. When configuring a cluster, it is important that this rule be followed.

##### 3.1.1.2 Global Credit and Borrowing

After all packets have been transmitted from a QOS queue, the queue's remaining total credit can not exceed the maximum credit allocated to that queue. Any credit bytes above the maximum credit limit are added to a global credit sum, and the total credit is set to the maximum credit.

Any queue may borrow from the global credit pool when doing so allows the queue to transmit an additional packet or is used to fill its allotted maximum credit level. This is done on a first come first served basis. The global credit system allows queues that are allocated less credit than necessary to saturate a device to make use of the additional bandwidth when it is not being used by the other QOS queues in the cluster.

Thus in the example above, the queue was set to 40Mb/s can use the entire bandwidth of the egress device when the other cluster queues are idle.

There is also a configurable maximum size on global credit. The limit on global credit is checked after every queue is processed. So for example, if the maximum global credit were set to zero, then the credit borrowing feature would be disabled.

### 3.1.1.3 QOS Queue Congestion and Packet Discard

A queue can become congested if the bandwidth of data arriving exceeds the bandwidth allocated or available. Each queue has a drop threshold expressed in bytes. Once the backlog in a QOS queue reaches its drop threshold, any packets that can not be transmitted are discarded until the backlog is cleared back below the threshold level.

For example, our 40Mb/s flow with the 20,000 byte burst could be assumed to be congested if more than one bursts worth of data we accumulated on the QOS queue. In this case, the drop threshold would be set to 40,000 bytes.

### 3.1.1.4 Destination Congestion and Credit Scaling

The destination queue for a QOS cluster may also be congested. For example, a cluster may configure 100Mb/s worth of data on an Ethernet device, but find that for various reasons that the device is only capable of sending 70Mb/s. The cluster algorithm will automatically scale the credit assigned to each queue according to how congested the egress queue becomes.

Each QOS cluster is configured with four egress congestion threshold values. Iteration credit is assigned to each queue in the cluster depending on the egress congestion, and the value of these four congestion thresholds. This is implemented as shown in the table below:

| Egress Queue Congestion (Backlog) Level           | QOS Queue Credit Assigned |
|---|---------------------------|
| Backlog < Threshold 1                             | Double credit             |
| Backlog >= Threshold 1, and Backlog < Threshold 2 | Normal credit             |
| Backlog >= Threshold 2, and Backlog < Threshold 3 | Half credit               |
| Backlog >= Threshold 3, and Backlog < Threshold 4 | Quarter credit            |
| Backlog >= Threshold 4                            | No credit                 |

Note that the use of double credit for near idle situations is used to ensure that each queue's burst potential be refilled as quickly as possible. It also allows the full bandwidth of a device to be used when the allocated bandwidth isn't quite enough to fill the device (for example allocating 98Mb/s from a 100Mb/s device).

If the egress queue for a cluster becomes congested due to external influences (like heavy load on the network), the credit scaling will affect each QOS queue equally. There may be cases where some flows require hard real time scheduling. In this case the queue can be marked as "real time" and exempt from credit scaling.

For example, in a 100Mb/s system that has two flows, a 40Mb/s flow and "everything else", the first queue in the cluster would be configured as 40Mb/s "real time", and the second queue can be configured as 60Mb/s (without the "real time" setting). As the available bandwidth on the network drops, the 40Mb/s flow would remain unaffected, while the 60Mb/s flow would be scaled down.

### 3.1.1.5 Fixed Priority Configuration

This algorithm can also be used to implement a fixed priority method, where each queue is serviced in a fixed priority with the first queue in the cluster being the highest priority. This is done by assigning all iteration credit to the first queue in the cluster, and setting the maximum credit of each queue to the maximum packet size. This guarantees that credit is only passed to subsequent queues when there are no packets waiting on the current queue.

For example, assume there are 3 queues, A, B, and C. In a simple priority system, queue A would always transmit packets when packets are available, while queue B only transmits when queue A is idle, and queue C only transmits when queue B is idle.

On a 100Mb/s system, the queues could be configured as follows:

Queue A

Iteration Credit = 313 (100 Mb/s is 312.5 bytes every 25us)

Max Credit = 1514

Queue B

Iteration Credit = 0

Max Credit = 1514

Queue C

Iteration Credit = 0

Max Credit = 1514

The way the algorithm works, queue A will get 313 bytes of credit at the start of each iteration. Since queue A can hold up to 1514 bytes as max credit, queue A will never pass credit onto queue B while queue A has a packet (if queue A has more than 1514 bytes of credit, it can always forward a packet).

Queue A must be idle for an entire packet time (1514 bytes of iteration credit) before any credit will start flowing into queue B. The same relationship holds between queue B and queue C. The only way queue B sends a packet is after queue A is idle for a packet time, and the only way queue C can send a packet is after queue B is idle for a packet time.

## 3.2 QOS Component Specification

The QOS PDSP is constructed around a simple method that allows for construction of complex QOS systems, or simple flow rated queues. The QOS hardware consists of a single PDSP with 64 ingress queues, and egress queues to the host CPU and physical peripherals.

The basic building block of the QOS system is a QOS queue. Each queue represents a flow priority, flow rate, drop policy, and egress queue. (Different QOS algorithms may or may not make use of all these properties.) Queues with the same egress queue are grouped together into a queue cluster called a QOS cluster. A cluster can contain from 1 to 9 QOS queues. Any of the 64 available QOS queues can be allocated into any given cluster QOS cluster, although a queue can not belong to more than one cluster.

### 3.2.1 QOS Queue

Each QOS queue is individually configured with its own iteration credit, maximum credit, and congestion threshold. There are also statistics for forwarded and dropped packets.

| Offset | Byte Field           |        |              |        |
|--------|----------------------|--------|--------------|--------|
|        | Byte 3               | Byte 2 | Byte 1       | Byte 0 |
| 0x0000 | Iteration Credit     |        | Egress Queue |        |
| 0x0004 | Total Credit         |        |              |        |
| 0x0008 | Maximum Credit       |        |              |        |
| 0x000C | Congestion Threshold |        |              |        |
| 0x0010 | Packets Forwarded    |        |              |        |
| 0x0014 | Packets Dropped      |        |              |        |

#### 3.2.1.1 QOS Queue Record Fields

| Name                 | Description  |
|----------------------|--|
| Egress Queue         | The Queue manger and Queue index of the forwarding queue.  |
| Iteration Credit     | The amount of forwarding byte "credit" that the queue receives every 25us.   |
| Total Credit         | The total amount of forwarding byte "credit" that that queue is currently holding.   |
| Maximum Credit       | The maximum amount of forwarding byte "credit" that the queue is allowed to hold at the end of the timer iteration. Any credit over the maximum limit is added to a global pool. |
| Congestion Threshold | The size in bytes at which point the QOS queue is considered to be congested. Setting this value to 0xFFFFFFFF will mean that the queue is never considered to be congested.     |
| Packets Forwarded    | The number of packets forwarded to the Egress Queue  |
| Packets Dropped      | The number of packets dropped due to congestion  |

### 3.2.2 QOS Cluster

The QOS cluster controls the order of how QOS queue are processed, and tracks properties of all QOS queues, like global credit and egress queue congestion.

| Offset | Byte Field                    |             |                |                    |
|--------|-------------------------------|-------------|----------------|--------------------|
|        | Byte 3                        | Byte 2      | Byte 1         | Byte 0             |
| 0x0000 | Global Credit                 |             |                |                    |
| 0x0004 | Maximum Global Credit         |             |                |                    |
| 0x0008 | QOS Queue 2                   | QOS Queue 1 | QOS Queue 0    | QOS Queue Count    |
| 0x000C | QOS Queue 6                   | QOS Queue 5 | QOS Queue 4    | QOS Queue 3        |
| 0x0010 | QOS Queue "Real Time" Flags   |             | QOS Queue 8    | QOS Queue 7        |
| 0x0014 | Egress Queue 0                |             | Flags          | Egress Queue Count |
| 0x0018 | Egress Queue 2                |             | Egress Queue 1 |                    |
| 0x001C | Egress Queue 4                |             | Egress Queue 3 |                    |
| 0x0020 | Egress Queue 6                |             | Egress Queue 5 |                    |
| 0x0024 | Egress Queue 8                |             | Egress Queue 7 |                    |
| 0x0028 | Egress Congestion Threshold 1 |             |                |                    |
| 0x002C | Egress Congestion Threshold 2 |             |                |                    |
| 0x0030 | Egress Congestion Threshold 3 |             |                |                    |
| 0x0034 | Egress Congestion Threshold 4 |             |                |                    |

#### 3.2.2.1 QOS Cluster Record Fields

| Name                          | Description  |
|-------------------------------|--|
| Global Credit                 | The amount of global credit available to the next QOS queue in the cluster   |
| Maximum Global Credit         | The maximum amount of global credit allowed to carry over to the next queue. Excess global credit is discarded.  |
| QOS Queue "Real Time" Flags   | This 9-bit mask contains 1 bit for each QOS queue in the cluster. When this bit is set for its corresponding QOS queue, iteration credit is treated as "real time" scheduling and does not scale when the egress queue become congested. |
| QOS Queue Count               | The number of QOS queues in the cluster (1 to 9)   |
| QOS Queue 0 ... 8             | The queue index (0 to 63) of each QOS queue in the cluster listed in priority order. These queue indices are relative to the configured QOS queue base index.  |
| Flags                         | Flags to control cluster options.<br>Bits 7:1 – Reserved (must be NULL)<br>Bit 0 – Round Robin Cluster Mode (must be set to 0)   |
| Egress Queue Count            | The total number of egress queues sampled to obtain the egress queue congestion estimation (1 to 9, but typically 1).  |
| Egress Queue 0 ... 8          | The Queue manger and Queue index of every egress queue enumerated in Egress Queue Count. These queue indices are absolute index values.  |
| Egress Congestion Threshold 1 | Egress Congestion Threshold point 1  |
| Egress Congestion Threshold 2 | Egress Congestion Threshold point 2  |
| Egress Congestion Threshold 3 | Egress Congestion Threshold point 3  |
| Egress Congestion Threshold 4 | Egress Congestion Threshold point 4  |

### 3.2.3 RR-MODE QOS Cluster

The QOS cluster in index 7 has an optional mode, different from other clusters. The fields are the same as a normal cluster, but they can be treated differently.

The purpose of the optional mode is to create two sets of four “round robin” queues. Each set will select packets in a round robin fashion. The set in queues 0-3 have strict priority over the set in queues 4-7. The entire cluster has a single egress queue and is timed using the iteration credit specified in the cluster. The following restrictions must also be true:

- This is an option for cluster index 7 only, and the Round Robin Cluster Mode flag must be set to 1 in the cluster Flags field as shown below.
- There are always 4 high priority queues and 4 low priority queues.
- The high priority queues are 56, 57, 58, and 59. The low priority queues are 60, 61, 62, and 63. These values are relative to the configured QOS queue base.
- The queue thresholds for the queue pending bits for the above 8 queues must be configured to be cleared when the queue is empty, and set when the queue is not empty.

| Original Name                 | Actual Use                            | Description   |
|-------------------------------|---------------------------------------|---|
| Global Credit                 | -                                     | The amount of global credit available to the next QOS queue in the cluster  |
| Maximum Global Credit         | -                                     | The maximum amount of global credit allowed to carry over to the next queue. Excess global credit is discarded.   |
| QOS Queue “Real Time” Flags   | Packet Size Adjustment                | This field holds the value of a packet size adjustment that can be applied to each packet. For example, setting this value to 24 can adjust for the preamble, inter-packet gap, and CRC for packets without CRC being sent over Ethernet. This adjustment value is applied across all queues. |
| QOS Queue Count               | -                                     | The number of QOS queues in the cluster. <b>It must be set to 8.</b>  |
| QOS Queue 0 ... 3             | High Priority Round Robin Queue Group | The queue index (0 to 63) of each QOS queue in the <i>high priority</i> “round robin” group. These queue indices are relative to the configured QOS queue base index. <b>These fields must be set to 56, 57, 58, and 59 respectively.</b>   |
| QOS Queue 4 ... 7             | Low Priority Round Robin Queue Group  | The queue index (0 to 63) of each QOS queue in the <i>low priority</i> “round robin” group. These queue indices are relative to the configured QOS queue base index. <b>These fields must be set to 60, 61, 62, and 63 respectively.</b>  |
| QOS Queue 8                   | -                                     | This field is ignored.  |
| Flags                         | -                                     | Flags to control cluster options.<br>Bits 7:1 – Reserved (must be NULL)<br>Bit 0 – Round Robin Cluster Mode (must be set to 1)  |
| Egress Queue Count            | -                                     | The total number of egress queues sampled to obtain the egress queue congestion estimation. <b>It must be set to 1.</b>   |
| Egress Queue 0                | -                                     | The Queue manger and Queue index of the egress queue used by the two round robin queue groups.  |
| Egress Queue 1 ... 8          | -                                     | These fields are ignored.   |
| Egress Congestion Threshold 1 | Iteration Credit                      | This is the “per timer tick” real time iteration credit for the cluster. ( <i>The iteration credit specified in each of the round robin queues is ignored.</i> )  |
| Egress Congestion Threshold 2 | Max Egress Backlog                    | This is the max number of bytes allowed to reside in the egress queue(s). Note that packets will be written until this threshold is crossed, so the actual number of bytes queued can be larger.  |
| Egress Congestion Threshold 3 | Queue Disable Mask                    | This 8-bit mask contains 1 bit for each QOS queue in the cluster. When this bit is set for its corresponding QOS queue, the queue is disabled for forwarding.   |
| Egress Congestion Threshold 4 | -                                     | This field is ignored.  |



### 3.3 SRIO Queue Monitoring

The QOS firmware includes a special SRIO queue monitoring mode. The firmware monitors a set of queues looking for Transmit packets, and moves them from the monitored queue to the actual transmit queue of the device. For each transmit packet moved, a global counter is incremented. Once the global counter reaches the programmable threshold for a particular transmit queue, that specific transmit queue is no longer serviced until the global count drops back below the threshold. The global queue is decremented when a packet arrives on one of the monitored TX completion queues. The monitored queues are called “shadow queues” as they are only a pre-staging to the actual final destination queue.

In addition to the transmit queues and transmit completion queues, six garbage collection queues are monitored. These queues may contain “transmit complete” packets for any of the monitored queues (plus potentially unmonitored queues). Any packet that arrives on one of these queues is checked for its original intended destination (by looking at the return queue index field the packet descriptor), and if intended for one of the monitored completion queues, the global count is decremented. Regardless, all packets are moved from the garbage collection shadow queues to the final garbage collection queues.

The entire queue set is collected into a single group with a single queue base. Hardware transmit queues are treated differently as they may not be run time configurable. The queue base must be a multiple of 32. The definition of the SRIO queues is as follows and is not configurable:

| Offset From Base Queue | Queue Usage                               |
|------------------------|---|
| 0                      | Shadow Garbage Collection Queue 0         |
| 1                      | Shadow Garbage Collection Queue 1         |
| 2                      | Shadow Garbage Collection Queue 2         |
| 3                      | Shadow Garbage Collection Queue 3         |
| 4                      | Shadow Garbage Collection Queue 4         |
| 5                      | Shadow Garbage Collection Queue 5         |
| 6                      | Shadow Transmit Queue (TXQ) 0             |
| 7                      | Shadow Transmit Queue (TXQ) 1             |
| 8                      | Shadow Transmit Queue (TXQ) 2             |
| 9                      | Shadow Transmit Queue (TXQ) 3             |
| 10                     | Shadow Transmit Queue (TXQ) 4             |
| 11                     | Shadow Transmit Completion Queue (TXCQ) 0 |
| 12                     | Shadow Transmit Completion Queue (TXCQ) 1 |
| 13                     | Shadow Transmit Completion Queue (TXCQ) 2 |
| 14                     | Shadow Transmit Completion Queue (TXCQ) 3 |
| 15                     | Shadow Transmit Completion Queue (TXCQ) 4 |
| 16                     | Transmit Completion Queue (TXCQ) 0        |
| 17                     | Transmit Completion Queue (TXCQ) 1        |
| 18                     | Transmit Completion Queue (TXCQ) 2        |
| 19                     | Transmit Completion Queue (TXCQ) 3        |
| 20                     | Transmit Completion Queue (TXCQ) 4        |

Note that the reserved queue index values can be used for other purposes and will be ignored by the firmware, but these empty slots may be used in future firmware.

#### 3.3.1 SRIO Queue Configuration

The SRIO Queue configuration is setup in PDSP scratch memory using the following format:

| Offset | Byte Field      |        |                 |        |
|--------|-----------------|--------|-----------------|--------|
|        | Byte 3          | Byte 2 | Byte 1          | Byte 0 |
| 0x0000 | <i>reserved</i> | SRIO   | SRIO Queue Base |        |

|        |                | Queue Count     |                            |
|--------|----------------|-----------------|----------------------------|
| 0x0004 | Hardware TXQ 0 | <i>reserved</i> | Threshold 0                |
| 0x0008 | Hardware TXQ 1 | <i>reserved</i> | Threshold 1                |
| 0x000C | Hardware TXQ 2 | <i>reserved</i> | Threshold 2                |
| 0x0010 | Hardware TXQ 3 | <i>reserved</i> | Threshold 3                |
| 0x0014 | Hardware TXQ 4 | <i>reserved</i> | Threshold 4                |
| 0x0018 | Reserved       |                 | Garbage Collection Queue 0 |
| 0x001C | Reserved       |                 | Garbage Collection Queue 1 |
| 0x0020 | Reserved       |                 | Garbage Collection Queue 2 |
| 0x0024 | Reserved       |                 | Garbage Collection Queue 3 |
| 0x0028 | Reserved       |                 | Garbage Collection Queue 4 |
| 0x002c | Reserved       |                 | Garbage Collection Queue 5 |

### 3.3.1.1 SRIO Queue Configuration Fields

| Name                       | Description   |
|----------------------------|---|
| SRIO Queue Base            | The Queue index of the base queue of the SRIO queue cluster. This value must be a multiple of 32.   |
| SRIO Queue Count           | The number of queues to monitor. This controls both the number of valid TXQ entries in this structure, plus the number of queues considered valid from the SRIO base queue index. |
| Threshold 0                | This is the high water mark for SRIO Queue Set 0 at which point additional transmit packets will not be accepted.   |
| Hardware TXQ 0             | This is the actual hardware queue onto which transmit packets must be eventually placed for Queue Set 0.  |
| Threshold 1                | This is the high water mark for SRIO Queue Set 1 at which point additional transmit packets will not be accepted.   |
| Hardware TXQ 1             | This is the actual hardware queue onto which transmit packets must be eventually placed for Queue Set 1.  |
| Threshold 2                | This is the high water mark for SRIO Queue Set 2 at which point additional transmit packets will not be accepted.   |
| Hardware TXQ 2             | This is the actual hardware queue onto which transmit packets must be eventually placed for Queue Set 2.  |
| Threshold 3                | This is the high water mark for SRIO Queue Set 3 at which point additional transmit packets will not be accepted.   |
| Hardware TXQ 3             | This is the actual hardware queue onto which transmit packets must be eventually placed for Queue Set 3.  |
| Threshold 4                | This is the high water mark for SRIO Queue Set 4 at which point additional transmit packets will not be accepted.   |
| Hardware TXQ 4             | This is the actual hardware queue onto which transmit packets must be eventually placed for Queue Set 4.  |
| Garbage Collection Queue 0 | This is the destination queue onto which packets are placed after they have been moved off the corresponding Garbage Collection Shadow Queue.                                     |
| Garbage Collection Queue 1 | This is the destination queue onto which packets are placed after they have been moved off the corresponding Garbage Collection Shadow Queue.                                     |
| Garbage Collection Queue 2 | This is the destination queue onto which packets are placed after they have been moved off the corresponding Garbage Collection Shadow Queue.                                     |
| Garbage Collection Queue 3 | This is the destination queue onto which packets are placed after they have been moved off the corresponding Garbage Collection Shadow Queue.                                     |
| Garbage Collection Queue 4 | This is the destination queue onto which packets are placed after they have been moved off the corresponding Garbage Collection Shadow Queue.                                     |
| Garbage Collection Queue 5 | This is the destination queue onto which packets are placed after they have been moved off the corresponding Garbage Collection Shadow Queue.                                     |

## 4. Firmware Command Interface

### 4.1 Firmware Command Handshake

#### 4.1.1 Command Handshake

The process of writing a command is to check to see if the command buffer is free, then write the command parameters, and finally write the command. Optionally, the caller can wait for command completion.

The command buffer is free when the “command” field of the first work in the command buffer is set to 0x00.

When a command is written, the host CPU must write the word containing the command byte \*last\*. The command buffer is in internal RAM and should not be marked as cacheable by the host CPU. If the RAM is cached on the host CPU, then the host must perform two separate writes and cache flushes; the first for writing the parameters, and then a second independent write and cache flush for writing the command word. All writes should be performed as 32 bit quantities.

Note that the first word of the command buffer appears in a noncontiguous memory region as the remaining fields in the buffer.

After the command is written, the PDSP will clear the “command” field upon command completion. The command results can then be read from the Return Code field.

#### 4.1.2 Command Buffer

The session router is programmed using a shared memory command buffer. The command buffer consists of a command word, followed by several parameters. The format of the buffer is as follows:

| Command Buffer Address | Field       |        |        |         |
|------------------------|-------------|--------|--------|---------|
|                        | Byte 3      | Byte 2 | Byte 1 | Byte 0  |
| 0x000B:C000            | Index       |        | Option | Command |
| 0x000B:C004            | Return Code |        |        |         |

The following is the breakdown of each field:

| Field       | Byte Width | Notes   |
|-------------|------------|---|
| Command     | 1          | QOS Command   |
| Option      | 1          | Command Option  |
| Index       | 2          | Command Index   |
| Return Code | 4          | Used to return status to the caller:<br>QCMD_RETCODE_SUCCESS 0x01<br>QCMD_RETCODE_INVALID_COMMAND 0x02<br>QCMD_RETCODE_INVALID_INDEX 0x03<br>QCMD_RETCODE_INVALID_OPTION 0x04 |

### 4.1.3 QOS Queue Region Base

Egress queues can be located anywhere in the system, but the QOS ingress queues are restricted to a set of 64 starting at a fixed base (which is a multiple of 32). Having a fixed base is not an issue since QOS queues must be allocated out of a general use pool in any case. However by using a base queue index, byte values (0-63) can be used to identify the individual QOS queues within clusters, making QOS structures more memory and performance efficient.

#### 4.1.3.1 QCMD\_GET\_QUEUE\_BASE

The QCMD\_GET\_QUEUE\_BASE command is used to read the queue number index of the base queue of the 64 QOS queue region.

Calling Parameters:

|         |                            |
|---------|----------------------------|
| Command | QCMD_GET_QUEUE_BASE (0x80) |
| Option  | not used                   |
| Index   | not used                   |

Returns:

|             |  |
|-------------|--|
| Index       | Queue index of the base queue of the 64 QOS queue region |
| Return Code | Success or Error Code                                    |

#### 4.1.3.2 QCMD\_SET\_QUEUE\_BASE

The QCMD\_SET\_QUEUE\_BASE command is used to set the queue number index of the base queue of the 64 QOS queue region.

Calling Parameters:

|         |  |
|---------|--|
| Command | QCMD_SET_QUEUE_BASE (0x81)                               |
| Option  | not used   |
| Index   | Queue index of the base queue of the 64 QOS queue region |

Returns:

|             |                       |
|-------------|-----------------------|
| Return Code | Success or Error Code |
|-------------|-----------------------|

#### 4.1.4 Timer Configuration

The PDSP timer determines when credit is passed out. The recommended interval is 25us. If the interval is set too low, the credit “resolution” becomes an issue (you don’t want to doll out one byte at a time), and the firmware performance may not be able to keep up with the interval requested.

The timer is configured by supplying a timer constant. The constant is computed as follows:

$$\text{Constant} = (\text{QMSS\_Clock\_Frequency} * \text{Desired\_Interval}) / 2$$

For example, if the QMSS is running at 350MHz (on Keystone I and II, it runs at DSP clock/3), and the desired credit interval is 25us, the constant value to program would be:

$$\text{Constant} = (350,000,000 * 0.000025) / 2 = 4375$$

##### 4.1.4.1 QCMD\_TIMER\_CONFIG

The QCMD\_TIMER\_CONFIG command is used to configure QOS credit interval timer.

Calling Parameters:

|         |                          |
|---------|--------------------------|
| Command | QCMD_TIMER_CONFIG (0x82) |
| Option  | not used                 |
| Index   | Timer Constant           |

Returns:

|             |                       |
|-------------|-----------------------|
| Return Code | Success or Error Code |
|-------------|-----------------------|

#### 4.1.5 Enable / Disable QOS Cluster

##### 4.1.5.1 QCMD\_CLUSTER\_ENABLE

The QCMD\_CLUSTER\_ENABLE command is used to enable or disable a QOS cluster. The cluster and queue configuration is performed by the host before executing this command. Only disabled clusters should be modified by the host processor.

When a cluster is disabled, all packets on QOS queues contained in that cluster are discarded.

Calling Parameters:

|         |   |
|---------|---|
| Command | QCMD_CLUSTER_ENABLE (0x83)  |
| Option  | Set to 1 to enable the cluster<br>Set to 0 to disable the cluster |
| Index   | QOS Cluster Index (0 to 7)  |

Returns:

|             |                       |
|-------------|-----------------------|
| Return Code | Success or Error Code |
|-------------|-----------------------|

## 4.1.6 Enable / Disable SRIO Queue Monitoring

### 4.1.6.1 QCMD\_SRIO\_ENABLE

The QCMD\_SRIO\_ENABLE command is used to enable or disable SRIO queue monitoring. Note that the host processor must setup the SRIO configuration structure in scratch RAM prior to enabling this mode.

Calling Parameters:

|         |   |
|---------|---|
| Command | QCMD_SRIO_ENABLE (0x84)   |
| Option  | Set to 1 to enable the SRIO queue monitoring<br>Set to 0 to disable the SRIO queue monitoring |
| Index   | not used  |

Returns:

|             |                       |
|-------------|-----------------------|
| Return Code | Success or Error Code |
|-------------|-----------------------|

## 4.2 Internal Memory Allocation

### 4.2.1 PDSP / QMSS Scratch RAM Allocation

The firmware assumes that 4K bytes of RAM are available, starting at 0x000B:C000.

| Address     | Length | Field  |
|-------------|--------|--|
| 0x000B:C000 | 0x40   | Command Buffer   |
| 0x000B:C040 | 0x1C0  | Eight QOS Cluster records (56 bytes each)                                |
| 0x000B:C200 | 0x600  | Sixty four QOS Queue records (24 bytes each)                             |
| 0x000B:CC00 | 0x30   | SRIO Queue Monitoring record (48 bytes)                                  |
| 0x000B:CC30 | 0x3C8  | -free-   |
| 0x000B:CFF8 | 0x004  | Magic number (0x80020000 for little endian or 0x80020001 for big endian) |
| 0x000B:CFFC | 0x004  | Firmware version number in format 0xV1.V2.V3.V4                          |